Assignment 1:

In this assignment, a non-functional java code is provided, which it should be refactored:

1. The java code should be refactored to functional way, vavr.io is preferred to be used.
2. Lombok library should be used for contractor, getters, and other methods.

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

class Person {
    private String name;
    private int age;

    Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    @Override
    public String toString() {
        return name + " (" + age + ")";
    }
}

public class NonFunctionalExample {

    public static void main(String[] args) {
        List<Person> people = createPeopleList();

        List<Person> adults = filterAdults(people);
        sortByName(adults);
        List<String> names = collectNames(adults);

        if (!names.isEmpty()) {
            printNames(names);
        } else {
            System.out.println("No names are there");
        }
    }

    private static List<Person> createPeopleList() {
        List<Person> people = new ArrayList<>();
        people.add(new Person("Alice", 23));
        people.add(new Person("Bob", 17));
        people.add(new Person("Charlie", 20));
        people.add(new Person("David", 15));
        people.add(new Person("Eve", 19));
        return people;
    }

    private static List<Person> filterAdults(List<Person> people) {
        List<Person> adults = new ArrayList<>();
        for (Person person : people) {
            if (person.getAge() >= 18) {
                adults.add(person);
            }
        }
        return adults;
    }

    private static void sortByName(List<Person> people) {
        Collections.sort(people, new Comparator<Person>() {
            @Override
            public int compare(Person p1, Person p2) {
                return p1.getName().compareTo(p2.getName());
            }
        });
    }

    private static List<String> collectNames(List<Person> people) {
        List<String> names = new ArrayList<>();
        for (Person person : people) {
            names.add(person.getName());
        }
        return names;
    }

    private static void printNames(List<String> names) {
        for (String name : names) {
            System.out.println(name);
        }
    }
}
```